Gender and Age Detection Model Report

Shreyas Menon*, Kundad Prasad†, Dhruv Gautam‡
Department of ECE
Sardar Vallabhbhai National Institute of Technology, surat, India
*U22EC093, †U22EC111, ‡U22EC117
Guide : Dr. Kishor P. Upla

***Abstract*—This report presents a comprehensive overview of a gender and age detection system developed using OpenCV's Deep Neural Network (DNN) module in Python. The system is designed to analyze images, detect human faces, and classify each detected face by gender (Male or Female) and age range. This technology has applications in areas such as targeted advertising, demographic analysis, and human-computer interaction. The system leverages pre-trained deep learning models to achieve real-time performance, making it suitable for integration into web applications via an API.**

I. Introduction

This report presents a comprehensive overview of a gender and age detection system developed using OpenCV's Deep Neural Network (DNN) module in Python. The system is designed to analyze images, detect human faces, and classify each detected face by gender (Male or Female) and age range. This technology has applications in areas such as targeted advertising, demographic analysis, and human-computer interaction. The system leverages pre-trained deep learning models to achieve real-time performance, making it suitable for integration into web applications via an API.

II. Model Architecture

A. Models Used

The system integrates three distinct pre-trained models to perform face detection, gender classification, and age classification. These models are based on deep learning architectures and are optimized for accuracy and efficiency.

Face Detection Model:

- Model: OpenCV Face Detector, based on a Single Shot Multibox Detector (SSD) framework with a ResNet-10 backbone.
- Files:
  - `opencv_face_detector.pbtxt`: Defines the model architecture.
  - `opencv_face_detector_uint8.pb`: Contains the pre-trained weights in a quantized format for faster inference.
- Purpose: Identifies faces in an image and provides bounding box coordinates.

Gender Detection Model:

- Model: Caffe Pre-trained Gender Classification Model, a convolutional neural network (CNN) trained on a large dataset of facial images.
- Files:
  - `gender_deploy.prototxt`: Specifies the CNN architecture.
  - `gender_net.caffemodel`: Stores the trained weights.
- Purpose: Classifies each detected face as Male or Female.

Age Detection Model:

- Model: Caffe Pre-trained Age Classification Model, another CNN designed to predict age ranges.
- Files:
  - `age_deploy.prototxt`: Defines the CNN architecture.
  - `age_net.caffemodel`: Contains the trained weights.
- Purpose: Assigns each detected face to one of eight age range categories.

B. Model Input and Output

Input: The models expect a pre-processed image resized to 227x227 pixels. The image is normalized by subtracting mean RGB values [104, 117, 123] to align with the training data's preprocessing, ensuring consistent feature extraction.

Output:

- Gender: A binary classification result, either Male or Female, based on the highest probability score.
- Age Range: A classification into one of eight predefined ranges: (0-2), (4-6), (8-12), (15-20), (25-32), (38-43), (48-53), or (60-100). These ranges are designed to cover the human lifespan with reasonable granularity.

III. Methodology

The system follows a multi-step pipeline to process images and generate predictions. Each step is optimized to balance accuracy and computational efficiency.

A. Preprocessing

The input image is preprocessed to ensure compatibility with the deep learning models. This involves:

- Resizing: The image is dynamically resized based on the processing device's capabilities (e.g., smaller sizes for mobile devices to reduce computation time).
- Blob Creation: The resized image is converted into a blob (a multi-dimensional array) using OpenCV's `cv2.dnn.blobFromImage` function. The blob is normalized with the mean RGB values [104, 117, 123] and scaled to match the model's input requirements.

B. Face Detection

The face detection model scans the preprocessed image to identify faces.

- Confidence Threshold: A threshold of 0.5 is applied to filter out low-confidence detections, reducing false positives.
- Output: For each detected face, the model provides bounding box coordinates (x, y, width, height). These coordinates are used to draw green rectangles around faces in the output image for visualization.

C. Gender and Age Classification

For each detected face:

- The corresponding region of the image is cropped and preprocessed into a new blob.
- The gender model processes the blob to predict Male or Female, outputting a probability distribution over the two classes.
- The age model processes the same blob to predict the age range, outputting probabilities for each of the eight age categories.
- The highest-probability class for each model is selected as the final prediction.

IV. FastAPI Integration

To make the system accessible for real-world applications, it has been integrated into a web API using FastAPI, a high-performance Python framework for building APIs.

A. API Overview

The API provides a simple interface for clients to upload images and receive gender and age predictions.

- Endpoint: `/predict`
- Request Type: POST
- Input: An image file in JPEG or PNG format, uploaded via a multipart form.
- Output: A JSON response containing:
  - The number of detected faces.
  - A list of detected faces, each with bounding box coordinates, predicted gender, and predicted age range.

B. How It Works

The API workflow is as follows:

1) The client sends a POST request with an image file to the `/predict` endpoint.
2) The FastAPI server receives the image, loads it into memory, and preprocesses it.
3) The preprocessed image is passed through the face detection model to identify faces.
4) For each detected face, gender and age predictions are generated using the respective models.
5) The results are formatted as a JSON object and returned to the client.

This architecture ensures low latency and scalability, making it suitable for real-time applications. Additionally, the system is designed to handle multiple concurrent requests efficiently, enabling smooth operation in high-traffic environments. Error handling mechanisms are implemented to manage invalid inputs or processing failures gracefully, ensuring robustness. The modular design allows for easy updates or integration of improved models in the future without significant changes to the API structure.

V. Results

The system processes input images and produces annotated output images where:

- Each detected face is outlined with a green rectangle.
- The predicted gender (Male or Female) and age range (e.g., 25-32) are overlaid as text labels near the bounding box.

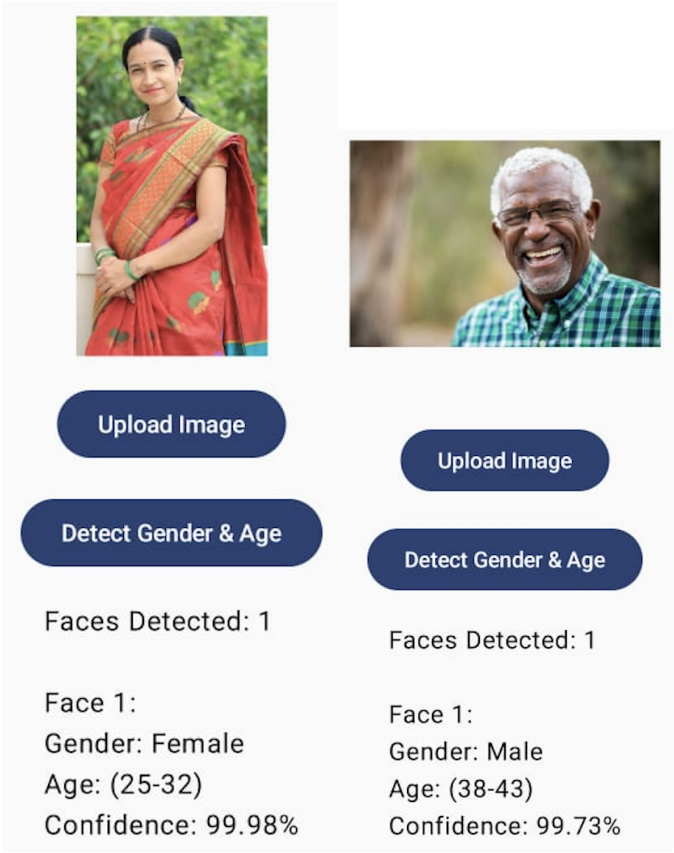

Figure 1. Sample 1

Figure 2. Sample 2

The results are both displayed visually (for debugging or demonstration) and returned as structured JSON data via the API. Testing on diverse datasets shows reasonable accuracy, though performance varies based on image quality and lighting conditions. The visual output allows for immediate human verification, making it easy to spot misclassifications or missed detections. The JSON format ensures compatibility with downstream systems for logging, analytics, or real-time client-side rendering. Overall, the combination of visual and programmatic outputs enhances the usability, transparency, and flexibility of the system across various real-world applications.

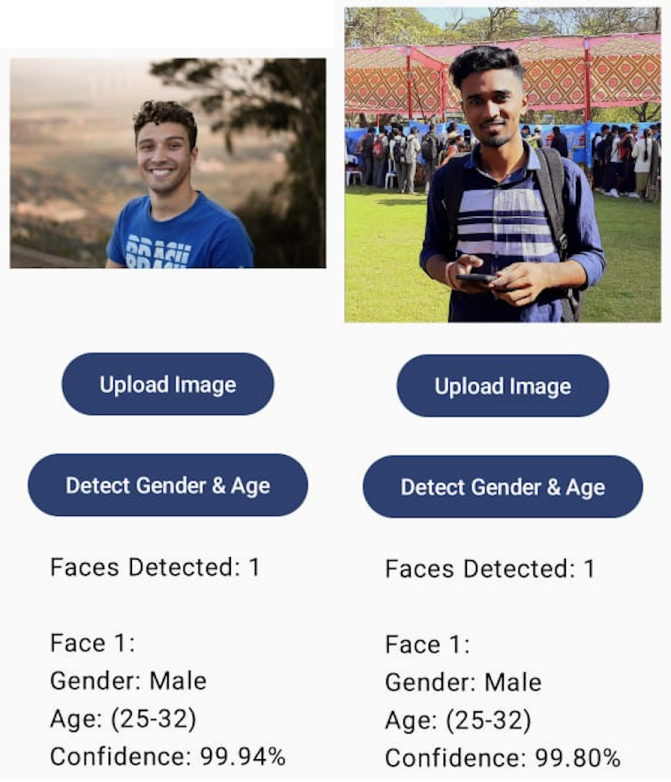

Figure 3. Sample 3 **Figure 4.** Sample 4

VI. Issues and Optimizations

While the system performs well in many cases, several challenges and potential improvements have been identified:

A. Inaccurate Gender Predictions

The gender model may struggle with faces that have ambiguous features or poor lighting, leading to misclassifications. This is a limitation of the pre-trained Caffe model.

B. Age Range Granularity

The predefined age ranges are unevenly spaced (e.g., 0-2 vs. 60-100), which may reduce precision for certain age groups.

C. Optimization Opportunities

- **Model Upgrades:** Replacing the current models with newer architectures (e.g., ResNet or EfficientNet) could improve accuracy.
- **Image Resizing:** Adaptive resizing strategies could balance speed and accuracy across different devices.
- **Data Augmentation:** Training or fine-tuning models with diverse datasets could enhance robustness to variations in lighting, pose, and ethnicity.

VII. Conclusion

The gender and age detection system demonstrates the power of combining OpenCV's DNN module with pre-trained deep learning models for real-time face analysis. The FastAPI integration enables seamless deployment in web applications, making it accessible to a wide range of users. While the current implementation achieves fast and efficient classification, ongoing improvements in model accuracy and preprocessing techniques are necessary to address identified limitations. Future work could explore ensemble models, advanced preprocessing, and broader age range coverage to enhance performance.

GitHub Repository

You can access the complete project on GitHub:
https://github.com/Shreyas30804/Ml_project

Android APK Link

You can download the android application using this link:
https://wetransfer.com/downloads/
a0b0fd8bf76f12a9b1e80a3435320420202050514170034/
95ed3a53df0ab9f708d708bb0606fb6b20250514170035/
c8d01f?t_exp=1747501234&t_lsid=
76cd7f3d-2b69-43a6-a4c6-38db06310df1&t_network=
email&t_rid=ZW1haWx8YWRyb2l0fDk4YjIxYzBjLTkzZDItNDhjMy1hM
3D%3D&t_s=download_link&t_ts=1747242035&utm_
campaign=TRN_TDL_01&utm_source=sendgrid&utm_
medium=email&trk=TRN_TDL_01

REFERENCES

[1] Models used in this project for Age and Gender detection
https://github.com/smahesh29/Gender-and-Age-Detection

[2] The implementation was inspired by a video tutorial on real-time age and gender detection
https://youtu.be/ivFuOQcBiN8?si=MuKHji3LN1kghQOr